Recent Developments in DAR and RefDBDAR

November, 20 , 2003

Natalia Ratnikova

DAR – Quick-n-Easy Applications Deployment

- Creates *application distribution (darball)* for the scram managed projects, based on the *run-time environment (rte)* of the available release:

  ```
  > dar   -c   <release topdir>   <tmp dir>
  ```

  - calls scram internally to get the rte,
  - creates and executes scripts to export the contents of the directories and files associated with the rte.
- Installs darball:

  ```
  > dar   -i   <darball>   <installation dir>
  ```

  - unpacks distribution,
  - customizes run-time environment setup scripts.

RefDB2DAR – Distributions Bookkeeping

- DAR interface to process RefDB requests:

```
> get_request 60
```

  - downloads request #60 from the RefDB to file FILE60 (example follows)

```
> python refdbdar  FILE60
```

  - parses and validates the RefDB request file
  - calls Packager: CMSIM_packager, CMKIN_packager, or DAR_packager for scram managed projects,
  - Packager builds executables according as requested and creates distribution

Scheme of Interactions

1. Production Coordinator fills web form to create DARball request. Generated request is stored in the RefDB, notification is sent by e-mail.
2. DARball is created then created using refdbdar and request file, based on software release installation at CERN.
3. Application is installed and tested in DAR runtime environment.
4. DARball is put into SRB for distribution and is ready for the production assignments.
5. Production sites get the assignments with the indication of the DARball (by name). DARball is then downloaded from the SRB and installed, using DAR, on the worker nodes.

DAR Problems and Solutions

- Distributions become very big:
  - runtime environment contains some superfluous directories and files. However for detection of files, that could be safely excluded, expert's knowledge of the software application is required.
  - a number of new expert options allow to filter the contents, but it may take several iterations to figure out what can be removed, and whether it is efficient and safe.
- Sometimes there is a need to include extra directories or files, not referred to by the default scram runtime environment:
  - scram allows to extend the runtime environment in the Project's config/Runtime file. Current refdbdar implementation uses this way to add extra requested geometry files, versions, and executables. It would be nice to foresee less awkward mechanism directly in DAR, that would help to satisfy such needs.

RefDB2DAR Problems and Solutions

- Currently no feedback from the refdbdar to RefDB :
  - it was originally foreseen, that refdbdar will submit status and metadata about completed request back to the RedDB
  - there is no clear “publishing” mechanism for produced DARballs, the status of DARball request stays “requested” even after distribution appears in the SRB and in the production assignments.
  - consequently, there is no warranty, that the distribution for a particular request is unique.
- Registering successfully completed request in the RefDB would allow to uniquely identify distributions based on the DARball metadata, e.g. the inventory list of contents plus the date of creation.

New Features in DAR

- Expert's options to reduce distribution size (*use with care!*):

  -f <ignore_file> : read list of “ignored” rte variable names from file: no files or directories associated with this variables will be included

  -e <exclude pattern> : list of file or directory names (wildcards supported) to be excluded from the distribution

- Replaces all duplicates in DARball by symbolic links
- Creates md5sum inventory list : one can now easily do consistency check in the installation directory: md5sum -c ./.DAR/Manifest.txt
- Other improvements:
  - more accurate handling of PATH-like variables; control available space during installation; control file persmissions; safe command execution in shell scripts; allow to redefine DAR runtime $PATH, ...

New Features in RefDB2DAR

- Extended request syntax to allow :
  - add multiple geometry files
  - add multiple XML geometry versions
  - specify list of ready executables to be included into distribution.
- Searches both through the environment set by scram, and in user's current $PATH.
- Passing exclude patterns to DAR through the configuration file .

Example of Request File (New Syntax)

```
Error=0
Application=OSCAR
Version=OSCAR_2_4_5
DARFileID=60
DARFileName=DAR_TEST2
DARStatus=Requested
NumberOfExecutables=18
ExecutablesList=FClistLFN;FCdeleteEntry;FCregisterPFN;FCpublish;FClistMetaData;
dsDump;FixColl;VerifyColl;ValidateRun;AttachRun;VerifyUC;FixUC;
mmencode;FCrenamePFN;FCregisterLFN;FClistPFN;oscar;FCBrowser.py
NumberOfGeometryFiles=2
GeometryFileName_0=cms132
GeometryFileSize_0=13729792
GeometryFileCksum_0=1079854733
GeometryFilePath_0=/afs/cern.ch/cms/oo/reconstruction/datafiles/cms132/cms132.rz
GeometryFileName_1=cms133
GeometryFileSize_1=13778944
GeometryFileCksum_1=3593014183
GeometryFilePath_1=/afs/cern.ch/cms/oo/reconstruction/datafiles/cms133/cms133.rz
NumberOfGeometryXMLversions=2
Geometry_PATH_0=/afs/cern.ch/cms/Releases/Geometry/Geometry_1_2_7/src
GeomPATHversion_0=Geometry_1_2_7
Geometry_PATH_1=/afs/cern.ch/cms/Releases/Geometry/Geometry_1_3_0/src
GeomPATHversion_1=Geometry_1_3_0
```

More About ...

- DAR development is moved to OCTOPUS
- All new features are available in the head revision
- DAR tool is now also available from the DPE pacman cache
- Old DAR versions can be used for installing new DARballs
- DAR documentation needs to be updated.
- Feedback is always welcome: questions, requests and bugs can be submitted through Savannah.
- Many thanks to Tony, Veronique and Julia and others for the contribution and feedback.